Introduction to Lambda Functions

Exercises

Copyright © James Raynard 2020

- Describe some of the disadvantages of using functors to provide predicate functions for algorithms
  - Boilerplate code (several lines to implement a one-line function)
  - Need to invent a name for it which won't conflict with anything else ("namespace pollution")
  - Interrupts writing and reading the code (productivity)

Copyright © James Raynard 2020

Page Two

- Explain briefly what is meant by a lambda expression
  - A lambda expression is an executable block of code that is written as a “local function” within another block of code
- Explain briefly how a lambda expression is implemented by the compiler
  - The compiler will generate a functor whose () operator contains the code from the body of the lambda expression
  - The compiler will create an instance of this functor and call it when required

Copyright © James Raynard 2020

- When creating a lambda expression, how do we enter
  - its name?
    - Lambda expressions are anonymous. We put [] where the name of the function would go
  - its arguments?
    - Inside (), just like a normal function

Copyright © James Raynard 2020

- When creating a lambda expression, how do we enter
  - its function body?
    - We write the function body inside {}, just like we would for an inline function
  - its return type?
    - This will be automatically deduced, provided the lambda function body is a single return statement (C++11)
    - This will be automatically deduced (C++14)

Copyright © James Raynard 2020

Page Four

- Rewrite the count_if example to use a lambda expression instead of the is_odd functor
- Write a simple program to test your code

Copyright © James Raynard 2020